REMARKS/ARGUMENTS

Claims have been amended to further clarify the subject matter regarded as the invention.

In the Office Action, the Examiner has rejected claims 1,3-10, 28 and 29 under 35 U.S.C. 102(e) as being anticipated by U.S. Patent Application Publication No. 2002/0147969 (*Lethin et al.*). The Examiner's rejections are traversed below for at least the following reasons.

a) *Lethin et al.* does NOT teach or suggest copying native code generated by an interpreter into a cache

It is noted U.S. Patent Application Publication No. 2002/0147969 (*Lethin et al.*) states that:

> If the correct branch is not in the cache, i.e. "no" in step S404, then flow proceeds to step S408 and one BRANCH_L1_RECORD (i.e. the record containing all fields which may be updated, such as encountered_sub_count and taken_sub_count) in the set designated by "S" above is removed from the L1 cache and written to the branch log. Next, the current branch information is written into the set designated by "S". Moreover, during writing of the current branch record into the set "S", the current branch record is placed as the first element of the set. This is because the same branch will very likely be executed again, thereby increasing performance and efficiency of the system. In other words sets S404 will be executed faster. Even when the branch is in the cache, i.e. "yes", it may be copied to the branch log if it has been executed a large number of times since it was last flushed. (Paragraph 168 of *Lethin et al.*)

However, it is respectfully submitted that step 408 described by *Lethin et al.* does NOT teach or suggest copying native code generated by an interpreter into a cache. This distinction is evident as Fig. 11 of *Lethin et al.* "illustrates a branch logging method" (Paragraph 157) that "counts how many times a branch has executed" (Paragraph 146).

Moreover, it is respectfully submitted that *Lethin et al.* does NOT teach or suggest this feature. Instead, *Lethin et al.* pertains to a "translation" system for "translation of target object code in the field of object code translators" when "it becomes necessary to convert object code which has been developed for one computer on another computer having a different computer architecture" (Paragraph 2). In other

words, the interpreter of *Lethin et al.* is "for individually translating object code into corresponding translated object code" (paragraph 11). As such, *Lethin et al.* states that interpreter can be further used for: "determining a number of executions of branch instructions in the source code, and a compiler [can be used] for grouping instructions of source object code into a segment when a number of executions of a corresponding branch instruction exceeds a threshold number, and for dynamically compiling the segment (Paragraph 11). Clearly, it is in the context of this translation of object code (not execution of virtual machine instructions) that the branch logging method depicted in Fig. 11 of used to "count how many times a branch has been executed" (Paragraph 146). However, it is respectfully submitted that *Lethin et al.* does NOT teach or suggest copying native code that has been effectively generated by an interpreter after execution of a virtual machine program instruction.

b) *Lethin et al.* does NOT teach or suggest: determining by an interpreter that executes virtual machine program instructions whether a basic block that includes native code corresponding to the virtual machine program instruction to be executed by the interpreter

Contrary to the Examiner's assertion, it is respectfully submitted that neither the interpreter of *Lethin et al.* executes virtual machine program instructions, nor does the L1 cache of *Lethin et al.* store native code corresponding to the virtual machine instruction. Again, the interpreter of *Lethin et al.* translates object code but it is not used for execution of virtual machine instructions. Furthermore, the cache of *Lethin et al.* is used to store counts of how many times a branch has been executed (Paragraph 146), but it is not used to store native code corresponding to a virtual machine program instruction.

c) *Lethin et al.* does NOT teach or suggest executing native code corresponding to a virtual machine program instruction from a cache

In view of the foregoing, it is respectfully submitted that *Lethin et al.* does NOT teach or suggest this additional feature.

CONCLUSION

Based on the foregoing, it is submitted that the claims are patentably distinct over the cited art of record. Additional limitations recited in the independent claims or the dependent claims are not further discussed because the limitations discussed above are sufficient to distinguish the claimed invention from the cited art. Accordingly, Applicant believes that all pending claims are allowable and respectfully requests a Notice of Allowance for this application from the Examiner.

Applicants hereby petition for an extension of time which may be required to maintain the pendency of this case, and any required fee for such extension or any further fee required in connection with the filing of this Amendment is to be charged to Deposit Account No. 500388 (Order No. SUN1P275). Should the Examiner believe that a telephone conference would expedite the prosecution of this application, the undersigned can be reached at the telephone number set out below.

Respectfully submitted,
BEYER WEAVER & THOMAS, LLP

/Rambusian/
R. Mahboubian
Reg. No. 44,890

P.O. Box 70250
Oakland, CA 94612-0250
(650) 961-8300